C. Obczyński

MAXIMA – tutorial

Ostatnia aktualizacja: 18 maja 2015 r.

Spis treści

1. Wprowadzenie

1.1. O programie Maxima

Maxima jest systemem komputerowym typu CAS (*Computer Algebra System*) wspierającym wykonywanie obliczeń symbolicznych. Jest on odpowiednikiem programów komercyjnych, takich jak Maple, Mathematica czy Matcad. Ma własny prosty interpreter, który przetwarza wszystkie wprowadzane przez nas polecenia.

Maxima wywodzi się z programu Macsyma opracowanego w latach sześćdziesiątych XX w. w Massachusetts Institute of Technology. Od 1982 roku program jest na darmowej licencji GPL Open Source.

Program Maxima umożliwia m.in.:

— przeprowadzanie podstawowych obliczeń arytmetycznych na liczbach;
— rozwiązywanie równań i nierówności: wielomianowych, wymiernych; trygonometrycznych i innych;
— definiowanie własnych funkcji i wyrażeń;
— wyznaczanie granic ciągów i funkcji;
— różniczkowanie i całkowanie symboliczne;
— sporządzanie wykresów funkcji na płaszczyźnie (2D) i w przestrzeni (3D);
— eksportowanie danych do formatów TEX i HTML.

Poniżej zamieściliśmy krótkie wprowadzenie do programu Maxima. Naszym celem nie było pisanie kolejnego podręcznika do tego programu, ponieważ takich opracowań jest już wiele w Internecie. Wszystkich zainteresowanych odsyłamy do poniższych, według nas ciekawych, stron:

— strona domowa programu Maxima, `http://maxima.sourceforge.net/` – pod linkiem `http://maxima.sourceforge.net/documentation.html` można znaleźć listę różnych tutoriali do Maximy;
— Introduction to Maxima for Economics – `http://statmath.wu.ac.at/courses/mvw_math1/aktuell/MaximaSkript.pdf`;
— Maxima w praktyce – `http://maxima.weilharter.info/documents/CAS_Maxima_Praxis.pdf`;
— Multivariable Calculus with Maxima – `http://people.ysu.edu/~gkerns/maxima/maximaintro/maximaintro.pdf`;
— Maxima (5.22.1) and the Calculus – `http://www.southernct.edu/~brin/papers/maxima_and_calculus.pdf`;
— Pakiety Matematyczne. Wprowadzenie do Maximy – `http://www-users.mat.umk.pl/~much/MK/samouczki/maximabook64x2_Chrzeszczyk.pdf`;
— Maxima – przewodnik praktyczny – `http://www.knf.ifd.uni.wroc.pl/materialy/maxima.pdf`;
— A Maxima-Gnuplot interface – `http://riotorto.users.sourceforge.net/gnuplot/`;
— A Maxima-VTK interface – `http://riotorto.users.sourceforge.net/vtk/index.html`;
— Syracuse Maxima – `http://melusine.eu.org/syracuse/maxima/`.

Na dołączonej do podręcznika płycie CD zamieściliśmy 4 pliki o rozszerzeniu `*.wxm` (każdemu rozdziałowi książki odpowiada jeden plik, np. ćwiczenia z rozdziału pierwszego zamieszczone

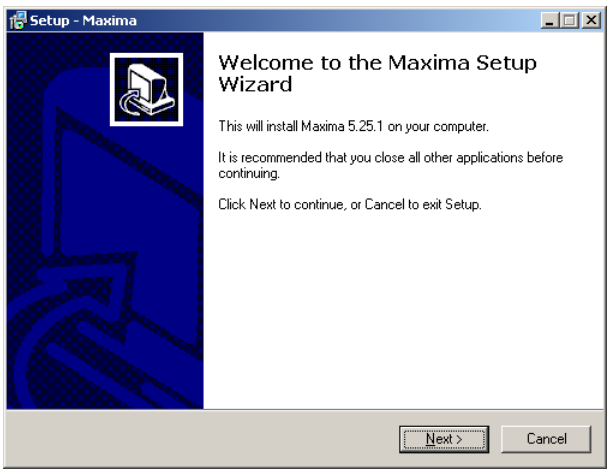
są w pliku `rozdzial1.wxm`), które można otworzyć w programie Maxima. Każdy z nich zawiera większość rozwiązań ćwiczeń z danego rozdziału książki wykonanych za pomocą poleceń zdefiniowanych w programie Maxima.

1.2. Instalacja programu Maxima

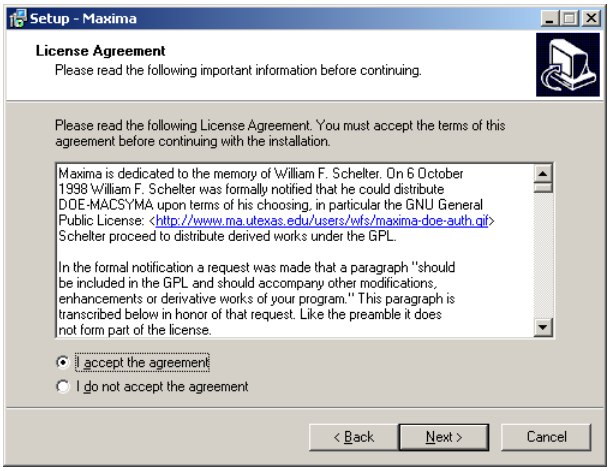
Aby pobrać program Maxima, wchodzimy na stronę

`http://sourceforge.net/projects/maxima/files/`

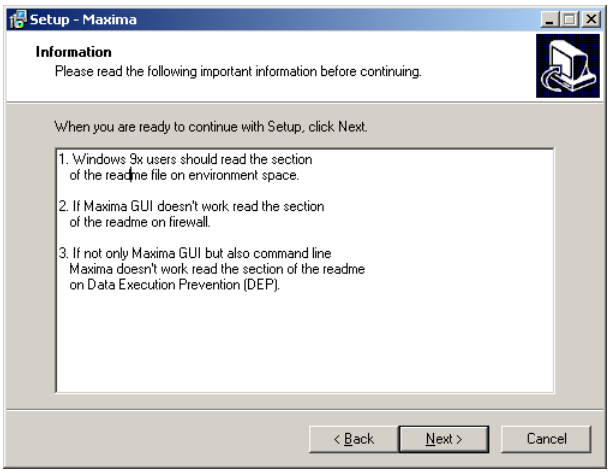
(instalacja znajduje się również na płycie dołączonej do książki) i wybieramy program instalacyjny Maximy odpowiedni do systemu operacyjnego, który mamy zainstalowany na naszym komputerze, tzn. Windows, Linux lub Mac OS (dostępne są również kody źródłowe do bezpośredniej kompilacji).
Po pobraniu pliku `maxima-5.25.1-gcl.exe` (dalsza część dotyczy Maximy w wersji 5.25.1 dedykowanej dla systemu operacyjnego Windows) klikamy na nim dwukrotnie, uruchamiając standardowy instalator programu Maxima.



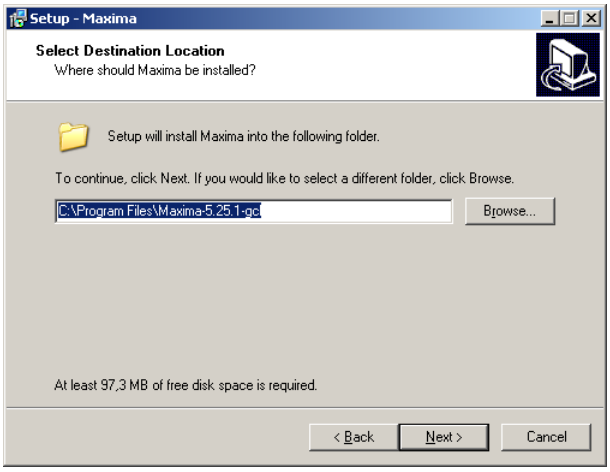
Wybierając Next, przechodzimy do następnego okna.



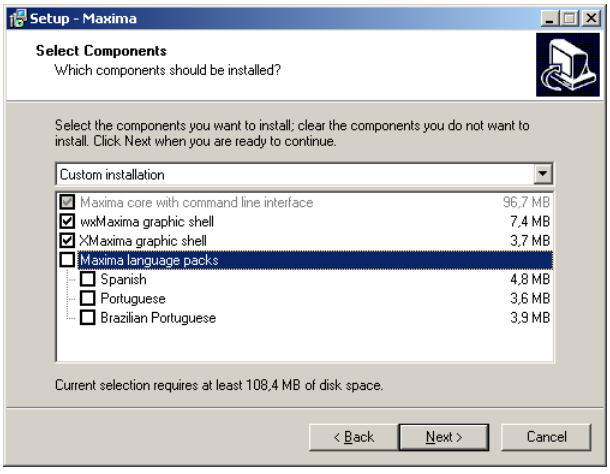
W tym miejscu musimy zaakceptować umowę licencyjną programu i kliknąć przycisk Next, aby przejść do dalszego etapu instalacji.



Zapoznajemy się z ogólnymi wytycznymi odnośnie do instalacji i klikamy przycisk Next.



Teraz wybieramy miejsce na dysku, gdzie zostanie zainstalowane oprogramowane Maximy, a następnie wybieramy Next.



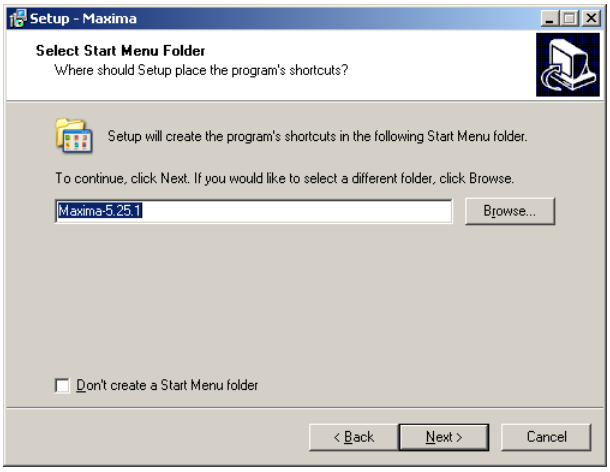
Wybieramy jeden z dostępnych trybów instalacji:

— *Full installation* (pełna instalacja),

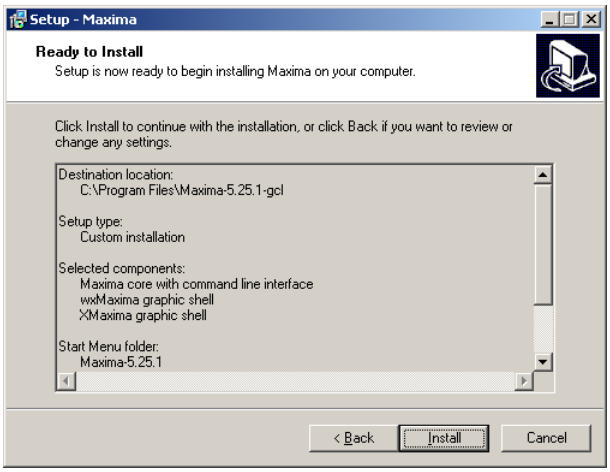
— *Compact installation* (minimalna instalacja),

— *Custom installation* (częściowa instalacja),

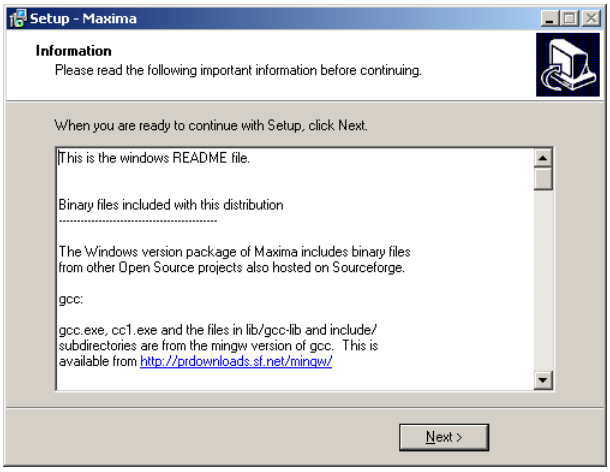
a następnie klikamy przycisk Next (my wybraliśmy opcję *Custom installation*, wyłączając dodatkowe pakiety językowe: Spanish, Portuguese i Brazilian Portuguese z grupy Maxima language packs).



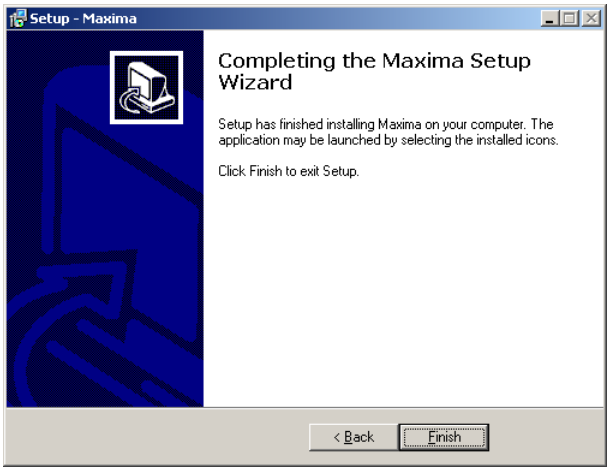
Po ustaleniu nazwy grupy (pojawi się ona w Menu Start i będzie zawierała wszystkie potrzebne składniki oprogramowania Maxima) klikamy Next.



Wybieramy jeszcze możliwość utworzenia skrótów do wxMaximy i Xmaximy, klikamy Next,

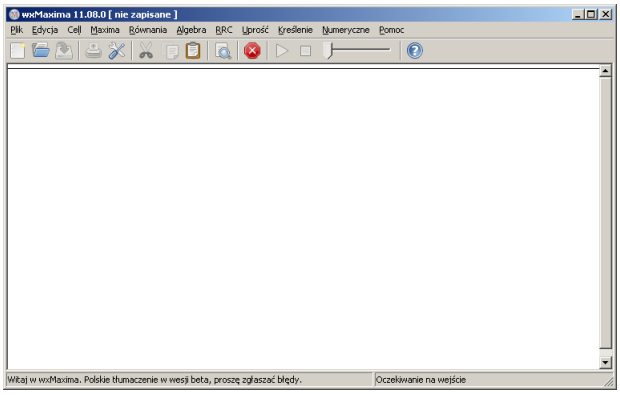


następnie Install i rozpoczynamy właściwy proces instalacji.



Jeżeli instalacja przebiegła poprawnie, ukaże się nam okno z informacją o zainstalowanym oprogramowaniu Maximy. Klikamy Next

i następnie Finish.
Po poprawnym zainstalowaniu Maximy i dwukrotnym kliknięciu na ikonie



(standardowo znajduje się ona na Pulpicie) ukaże nam się okno tego programu. Główne okno Maximy, jak każde standardowe okno systemu Windows, składa się z paska menu (podzielonego na grupy poleceń do wyboru), paska narzędziowego z takimi opcjami, jak kopiuj, wklej, wytnij, głównego okna programu i paska stanu.

1.3. Jak zacząć pracę z programem Maxima?

Po uruchomieniu Maximy otworzy nam się nowy plik, który zapisujemy (Plik/Zapisz jako) pod wybraną przez nas nazwą (w naszym przypadku jest to rozdzial00). Plik automatycznie dostaje rozszerzenie `*.wxm`.
Będąc w obrębie głównego okna, wprowadzamy kolejno polecenia, pamiętając, aby każde z nich kończyło się znakiem średnika `;`. W celu wykonania polecenia stosujemy kombinację klawiszy Shift ⇑ + Enter . Przykładowo, aby znaleźć wynik dodawania liczb `12` i `65`, wprowadzamy

```
12+65;
```

i wciskamy kombinację klawiszy Shift ⇑ + Enter . Dostajemy

```
(%i1)  12+65;
(%o1)
```

$$77$$

Chcąc znaleźć wynik mnożenia liczb `123` i `456`, wprowadzamy `123*456;` i wciskamy kombinację klawiszy Shift ⇑ + Enter . Dostajemy

```
(%i2)  123*456;
(%o2)
```

$$56088$$

Zauważmy, że każde polecenie w Maximie jest numerowe, przy czym wiersze wejściowe są oznaczone: `%i1`, `%i2`, `%i3` itd. (gdzie `i` jest skrótem od `input`), a wiersze wyjściowe są oznaczone: `%o1`, `%o2`, `%o3` itd. (gdzie `o` jest skrótem od `output`). Maxima umożliwia odwołanie do ostatniego wyniku poprzez użycie symbolu `%`. Na przykład, wpisanie

```
%+44;
```

i wciśnięcie kombinacji klawiszy Shift ⇑ + Enter daje

```
(%i3)  %+44;
(%o3)
```

$$56132$$

i jest to inny sposób wykonania polecenia

```
123*456+44;
```

(w tym wypadku `%` reprezentuje liczbę 56088). W Maximie mamy również możliwość odwołania do jednego z poprzednich wyników poprzez wykonanie polecenia `%k` (`k` to numer wybranego wyniku). Na przykład, wykonanie polecenia

```
%1+%2+%3;
```

doda do siebie wszystkie wyniki zapamiętane w wierszach wyjściowych `%1`, `%2` oraz `%3`, tzn.

```
(%i4)  %o1+%o2+%o3;
(%o4)
```

$$112297$$

Jeżeli obliczenia przeprowadzamy na liczbach niewymiernych, to wyniki często otrzymujemy w formie symbolicznej

```
(%i5)  sqrt(2);
(%o5)
```

$$\sqrt{2}$$

Jeżeli wynik obliczeń chcemy wyrazić w postaci dziesiętnej (przybliżonej), to możemy zastosować polecenie `numer`

```
(%i6)  %,numer;
(%o6)
```

$$1.414213562373095$$

lub polecenie `float`

```
(%i7)  float(%o6);
(%o7)
```

$$1.414213562373095$$

Otrzymaliśmy wynik zapisany z dokładnością do 16 cyfr znaczących. Chcąc zwiększyć dokładność obliczeń, stosujemy polecenie `fpprec`; wykonując polecenie

```
(%i8)  fpprec:30;
(%o8)
```

$$30$$

ustawiamy precyzję obliczeń na 30 cyfr znaczących. Stosując teraz polecenie `bfloat`, dostajemy wartość wyrażenia $\sqrt{2}$ obliczoną z dokładnością do 30 cyfr znaczących zapisaną w notacji naukowej

```
(%i9)  bfloat(%o6);
(%o9)
```

$$1.41421356237309514547462185874_B \times 10^0$$

tzn. zapis `1.41421356237309504880168872421b0` jest tym samym, co zapis

$$1.41421356237309504880168872421 * 10^0$$

(wynikiem obliczeń jest zatem liczba 1.41421356237309504880168872421).
W celu wykonania polecenia bez pokazywania efektu, tzn. bez wyświetlania wiersza out, na końcu polecenia stosujemy znak dolara `$`. Na przykład, stosując polecenie `fprec:40`, ustalamy stałą

dokładność obliczeń na 40 cyfr znaczących, nie pokazując wiersza wyjściowego out z wynikiem wykonania tego polecenia

```
(%i10)  fprec:40$
```

Komentarze w programie Maxima wprowadzamy, używając znaków `/*` oraz `*/`

```
(%i11)  /* Dodawanie dwóch liczb */ 4+5;
(%o11)
```

$$9$$

Zauważmy, że wykasowanie wybranego wiersza wejściowego (pewnego inputu) lub wykonanie wiersza wejściowego kilkukrotnie zaburza kolejną numerację wierszy wejściowych – nie przeszkadza to jednak w niczym w przeprowadzanych obliczeniach. Więcej poleceń omówimy przy rozwiązywaniu wybranych zadań z rozdz. 1–4 naszej książki *Granice i pochodne. Metody rozwiązywania zadań*. We wszystkich zadaniach korzystamy z Maximy w wersji 5.25.1 pracującej pod kontrolą systemu operacyjnego Windows.

2. Podstawy Maximy

Przejdźmy teraz do krótkiego omówienia podstawowych poleceń używanych w Maximie.

2.1. Podstawowe operatory matematyczne

Poniżej przedstawiamy listę podstawowych operatorów arytmetycznych używanych w Maximie:

`+`	operator dodawania
`-`	operator odejmowania
`/`	operator dzielenia
`*`	operator mnożenia
`^`	operator potęgowania

Przykładowo:

```
(%i12)  2+4;
(%o12)
```

$$6$$

```
(%i13)  (2*8/4)^2;
(%o13)
```

$$16$$

```
(%i14)  (100/4)^3 - (24/4^2)^3;
(%o14)
```

$$\frac{124973}{8}$$

2.2. Stałe

Poniżej przedstawiamy listę podstawowych stałych używanych w Maximie:

`%e`	stała Eulera $e = 2,71828\ldots$
`%pi`	liczba $\pi = 3,14159\ldots$
`%i`	jednostka urojona $i = \sqrt{-1}$
`inf`	nieskończoność ∞
`minf`	minus nieskończoność $-\infty$

Przykładowo, liczbę e w Maximie możemy zapisać następująco:

```
(%i15)  %e;
(%o15)
```

$$e$$

Przybliżona wartość tej liczby wynosi

```
(%i16)  float(%e);
(%o16)
```

$$2.718281828459045$$

2.3. Podstawowe funkcje

Podstawowe funkcje stosowane w matematyce można znaleźć w Maximie:

`abs(x)`	wartość bezwzględna liczby x
`entier(x)`	całość liczby x
`sqrt(x)`	pierwiastek kwadratowy liczby x
`log(x)`	logarytm naturalny liczby x
`exp(x)`	exponent liczby x
`sin(x)`	sinus liczby x
`cos(x)`	cosinus liczby x
`tan(x)`	tangens liczby x
`cot(x)`	cotangens liczby x

Przykładowo, chcąc poznać wartość wyrażenia

$$|-4|+|-10|+\left(\sqrt{16}\right)^2+\cos(\pi)$$

wystarczy napisać

```
(%i17)  abs(-4) + abs(-10) + (sqrt(16))^2 + cos(%pi);
(%o17)
```

$$29$$

2.4. Zmienne i funkcje użytkownika

Wyrażenia, tak jak liczby, mogą być przechowywane w zmiennych. Nazwa zmiennej może składać się ze znaków: `A` do `Z`, `a` do `z`, `0` do `9`, `%` oraz _. Przykłady poprawnych nazw zmiennych to: `a`, `b1`, `s_1`.

`:`	operator przypisania
`:=`	operator definiowania funkcji
`kill(x)`	usuwa zmienną `x` z pamięci
`kill(all)`	usuwa wszystkie zmienne oraz funkcje z pamięci

Wartość do zmiennej przypisuje się za pomocą operatora przypisania `:`. Operator ten przypisuje wartość po prawej strony wyrażenia do zmiennej po lewej stronie.
Przykładowo, chcąc przypisać zmiennej x wartość 4 musimy zapisać

```
(%i18)  x:4 /* od tego miejsca zmienna x ma wartość 4 */;
(%o18)
```

$$4$$

Nazwą tej zmiennej możemy operować w dalszych obliczeniach, na przykład

```
(%i19)  y:6 + x /* zmienna y=6+4=10 */;
(%o19)
```

$$10$$

Maxima umożliwia także stworzenie własnych funkcji za pomocą operatora `:=`. Po nazwie funkcji piszemy zawsze nawiasy `()`, które zawierają listę argumentów (zmiennych) oddzielonych od siebie przecinkami.
W poniższym przykładzie zdefiniujemy funkcję liniową $f(x) = x + 1$.

```
(%i20)  f(x):=x+1;
(%o20)
```

$$f\left(x\right) := x + 1$$

Teraz, chcąc wyznaczyć wartość funkcji f dla argumentu $x = 10$, napiszemy

```
(%i21)  f(10);
(%o21)
```

$$11$$

W Maximie zmienne oraz funkcje istnieją dopóty, dopóki nie zostanie zamknięta sesja Maximy. Czasami przydatne jest usunięcie tych zmiennych lub funkcji wcześniej. Dokonujemy tego za pomocą polecenia `kill()`. Przykładowo

```
(%i22)  kill(x);
(%o22)
```

$$\textbf{done}$$

Od tego momentu zmienna x nie będzie miała przypisanej żadnej wartości

```
(%i23)  x;
(%o23)
```

$$x$$

Wszystkie zmienne możemy wyczyścić z pamięci za pomocą polecenia `kill(all)`.

2.5. Operatory logiczne

Lista podstawowych operatorów logicznych w Maximie to:

`=`	operator porównania
`is(x)`	operator logiczny ustalający, czy `x` jest `true`, czy `false`
`equal(x,y)`	operator zwracający `true` (`false`), jeżeli x i y (nie) są równe
`#`	negacja operatora równości
`<`	operator mniejsze niż
`<=`	operator mniejsze niż lub równe
`>`	operator większe niż
`>=`	operator większe niż lub równe
`and`	operator koniunkcji
`or`	operator alternatywy
`not`	operator negacji
`true`	stwierdzenie prawdziwe
`false`	stwierdzenie fałszywe
`unknown`	stwierdzenie o nieznanej logicznej wartości

Niech

```
(%i26)  x:5;
(%o26)
```

$$5$$

```
(%i27)  y:5;
(%o27)
```

$$5$$

Jeżeli chcemy sprawdzić, czy wartości zmiennych x i y są równe, możemy wydać polecenie

```
(%i15)  is(equal(x,y));
(%o15)
```

true

Widzimy więc, że wartości zmiennych x i y są równe. Podobnie postępujemy z pozostałymi operatorami logicznymi. Na przykład

```
(%i16)  is(x > 10) /* czy x jest większe od 10*/;
(%o16)
```

false

Oczywiście nie.

2.6. Rozwijanie, rozkładanie oraz upraszczanie wyrażeń

Mimo istnienia pewnych algorytmów dla standardowych procedur, upraszczanie wyrażeń (algebraicznych, trygonometrycznych, logarytmicznych i in.) jest nadal jednymi z trudniejszych zadań dla systemów algebry komputerowej. Trudność leży w wyborze, kiedy i którą procedurę wybrać. Przykładowo, niektóre matematyczne sytuacje wymagają rozkładu na czynniki, inne natomiast wymagają, byśmy najpierw rozwinęli dane wyrażenie, a następnie upraszczali. Systemy algebry komputerowej, do których zalicza się Maxima, nie mają możliwości określenia, którą z dróg wybrać najpierw. Dlatego małą ilość uproszczeń da się wykonać automatycznie – do wielu wyrażeń można zastosować więcej niż jedno polecenie upraszczające.

`ratsimp(expr)`	upraszcza wyrażenie
`fullratsimp(expr)`	wielokrotnie upraszcza wyrażenie
`rootscontract(expr)`	zamienia iloczyn pierwiastków na pierwiastek iloczynu
`radcan(expr)`	upraszcza wyrażenia logarytmiczne, wykładnicze i pierwiastkowe
`logcontract(expr)`	składa wiele funkcji logarytmicznych w pojedyncze logarytmy
`expand(expr)`	rozwija wyrażenie
`ratexpand(expr)`	rozwija wyrażenie (polecenie na ogół stosowane do wielomianów)
`factor(expr)`	rozkłada wyrażenie na czynniki
`trigsimp(expr)`	upraszcza wyrażenia trygonometryczne (wykorzystuje związek $\sin^2 x + \cos^2 x = 1$)
`trigexpand(expr)`	rozkłada funkcje trygonometryczne sumy oraz iloczynu kątów (może wymagać wielokrotnego użycia tego polecenia)
`trigreduce(expr)`	składa sumy oraz potęgi sinusów i cosinusów w sinusy i cosinusy wielokrotności ich argumentów
`trigrat(expr)`	upraszcza (do postaci kanonicznej) wyrażenie trygonomoetryczne

Przykładowo, dla wyrażenia

```
(%i19)  (x^2-1)/(x-1);
(%o19)
```

$$\frac{x^2-1}{x-1}$$

mamy

```
(%i20)  ratsimp((x^2-1)/(x-1));
(%o20)
```

$$x+1$$

Użycie pozostałych poleceń prezentujemy poniżej.

```
(%i21)  '((x-1)/(sqrt(x)-1)) = radcan((x-1)/(sqrt(x)-1));
(%o21)
```

$$\frac{x-1}{\sqrt{x}-1} = \sqrt{x}+1$$

```
(%i22)  expand((a+1)^2);
```

```
(%o22)
```

$$a^2 + 2\,a + 1$$

```
(%i23)  factor(a^2-5*a+6);
(%o23)
```

$$(a - 3)\,(a - 2)$$

```
(%i24)  '((cos(x)+sin(x))^2) = trigsimp((cos(x)+sin(x))^2);
(%o24)
```

$$(\sin x + \cos x)^2 = 2\,\cos x\,\sin x + 1$$

```
(%i25)  trigreduce(sin(x)^3*cos(x));
(%o25)
```

$$\frac{2\,\sin\left(2\,x\right) - \sin\left(4\,x\right)}{8}$$

2.7. Pakiety

Tak jak w wielu innych środowiskach obliczeniowych, Maxima dostarcza zbioru podstawowych poleceń oraz duży wybór pakietów dodatkowych. Polecenie

```
load(pakiet);
```

ładuje zawartość pliku `pakiet`. Na przykład pakiet `solve_rat_ineq` wczytujemy za pomocą polecenia

```
(%i32)  load(solve_rat_ineq)$
```

2.8. Równania

Podstawowe polecenia do rozwiązywania równań:

`solve(eqn,x)`	rozwiązuje równanie `eqn` względem zmiennej x
`solve([eqn1,eqn2],[x,y])`	rozwiązuje układ równań `eqn1` i `eqn2` względem zmiennych x i y
`lhs(eqn)`	odczytuje lewą stronę równania `eqn`
`rhs(eqn)`	odczytuje prawą stronę równania `eqn`
`realroots(poly)`	znajduje pierwiastki rzeczywiste wielomianu `poly`
`allroots(poly)`	znajduje numeryczne przybliżenie pierwiastków wielomianu `poly`
`find_root(eqn,x,x0,x1)`	znajduje przybliżone rozwiązanie równania `eqn` na przedziale $[x_0, x_1]$

Aby rozwiązać równanie kwadratowe $x^2 - 5x + 6 = 0$, wydamy komendę

```
(%i27)  solve(x^2 - 5*x + 6 = 0,x);
(%o27)
```

$$[x = 3, x = 2]$$

Definiując równanie

```
(%i28)  row:2*x + 4 = -x + 13;
(%o28)
```

$$2\,x + 4 = 13 - x$$

możemy je rozwiązać

```
(%i29)  solve(row,x);
(%o29)
```

$$[x = 3]$$

bądź wykonywać różne przekształcenia tylko na prawej bądź lewej stronie tego równania, na przykład

```
(%i30)  lhs(row) - 2*x - 4;
(%o30)
```

$$0$$

Szukając rozwiązań równania $x^3 + 1 = 0$ (czyli pierwiastków wielomianu $x^3 + 1$), stosujemy polecenie `solve`

```
(%i31)  solve(x^3 + 1=0,x);
(%o31)
```

$$\left[x = -\frac{\sqrt{3}\,i - 1}{2}, x = \frac{\sqrt{3}\,i + 1}{2}, x = -1\right]$$

które znajdzie wszystkie pierwiastki (rzeczywiste i zespolone) danego równania. Polecenie `realroots(poly)`

```
(%i32)  realroots(x^3 + 1=0);
(%o32)
```

$$[x = -1]$$

znajdzie tylko pierwiastki rzeczywiste tego wielomianu.

Pakiet `solve_rat_ineq` pozwala na rozwiązywanie przez Maximę podstawowych nierówności.

```
(%i33)  solve_rat_ineq(x^2-5*x+6 >=0);
(%o33)
```

$$[[x \leq 2], [x \geq 3]]$$

Kolejny pakiet `to_poly_solver` jest odpowiedzialny za rozwiązywanie równań. Po jego załadowaniu

```
(%i34)  load(to_poly_solver)$
    to˙poly˙solve: to˙poly˙solver.mac is obsolete; I'm loading to˙poly˙solve.mac instead.
```

do rozwiązywania równań stosować polecenie `%solve`. Na przykład

```
(%i35)  %solve(abs(x+1)=1,x);
(%o35)
```

$$[x = -2] \cup ([x = 0])$$

Maxima jest narzędziem, które ma wspierać nasze wyliczenia, a nie tylko podawać suche wyniki. Często więc stosujemy polecenia, dzięki którym możemy śledzić poprawoność kroków naszych rozwiązań. Przykładem takiego polecenia jest `subst(a,b,wyr)`, czyli podstaw `a` za `b` w wyrażeniu `wyr`. Metodę podstawiania wykorzystujemy w wielu typach zadań. Na przykład, aby rozwiązać równanie dwukwadratowe

```
(%i2)  row:x^4-5*x^2+6=0;
(%o2)
```

$$x^4 - 5\,x^2 + 6 = 0$$

zastosujemy metodę podstawiania. Zastosujemy podstawienie $x^2 = t$.

```
(%i3)  subst(t,x^2,row);
(%o3)
```

$$x^4 - 5\,t + 6 = 0$$

Nie takiego efektu oczekiwaliśmy. Domyślnie Maxima podstawia nowe zmienne za wyrażenia, które są jawnie podane, w tym przypadku za wyrażenie x^2. Aby zmusić program do podstawienia $x^4 = t^2$ musimy zmienić domyślne ustawienia opcjonalnej wartości (flagi) `exptsubst`, która jest ustawiona na wartość `false`.

```
(%i4)  exptsubst: not exptsubst;
(%o4)
```

true

Wówczas wywołując ponownie

```
(%i5)  subst(t,x^2,row);
(%o5)
```

$$t^2 - 5\,t + 6 = 0$$

uzyskamy równanie kwadratowe ze względu na zmienną t.

2.9. Granice i pochodne

`limit(fun,x,pkt)`	oblicza granicę funkcji `fun` zmiennej x w punkcie `pkt`
`limit(fun,x,pkt,minus)`	oblicza granicę lewostronną funkcji `fun` zmiennej x w punkcie `pkt`
`limit(fun,x,pkt,plus)`	oblicza granicę prawostronną funkcji `fun` zmiennej x w punkcie `pkt`

`limit(fun,x,inf)`	oblicza granicę funkcji `fun` zmiennej x w plus nieskończoności
`limit(fun,x,-inf)`	oblicza granicę funkcji `fun` zmiennej x w minus nieskończoności

```
(%i6)  'limit((x^2-3)/(x+4),x,1)=limit((x^2-3)/(x+4),x,1);
(%o6)
```

$$\lim_{x \to 1} \frac{x^2-3}{x+4} = -\frac{2}{5}$$

`diff(fun,x)`	oblicza pierwszą pochodną funkcji `fun` zmiennej x
`diff(fun,x,n)`	oblicza n-tą pochodną funkcji `fun` zmiennej x

```
(%i7)  g(x):=x^2;
(%o7)
```

$$g\left(x\right) := x^2$$

```
(%i8)  diff(g(x),x);
(%o8)
```

$$2\,x$$

```
(%i9)  dg(x):=diff(g(x),x);
(%o9)
```

$$\mathrm{dg}\left(x\right) := \mathrm{diff}\left(g\left(x\right), x\right)$$

```
(%i10)  dg(x);
(%o10)
```

$$2\,x$$

```
(%i11)  dg(4);
   ** error while printing error message **
   ~:M: second argument must be a variable; found ~M
   #0: dg(x=4)
    – an error. To debug this try: debugmode(true);
```

```
(%i12)  define(dg(x),diff(g(x),x));
(%o12)
```

$$\mathrm{dg}\,(x) := 2\,x$$

```
(%i13)  dg(4);
(%o13)
```

$$8$$

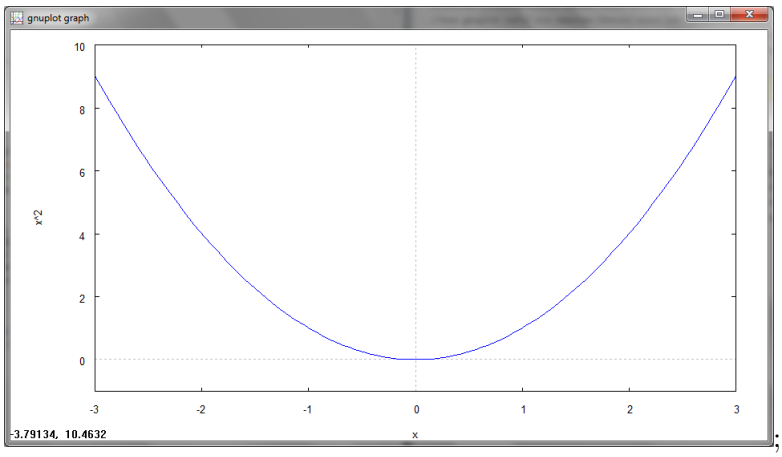
2.10. Wykresy

Rysowanie wykresów funkcji jednej bądź dwu zmiennych jest ważnym zadaniem do oceniania zachowania się funkcji. Maxima używa zewnętrznych pakietów do rysowania wykresów. Istnieje możliwość wyboru między dwoma takimi pakietami: gnuplot (domyślny) – zobacz `http://www.gnuplot.info/` oraz `xmaxima` (dawniej znana jako openmath).
Standardowe polecenia do rysowania wykresów w Maximie to `plot2d`, `plot3d`. Do wyświetlania obrazów w głównym oknie (inline) stosuje się funkcję `wxplot2d` oraz `wxplot3d`.
Polecenie

```
plot2d(f(x),[x,x_min,x_max], opt1, opt2, ...)
```

jest jednym z najczęściej stosowanych poleceń do wykreślania funkcji jednej zmiennej postaci $y = f(x)$. Za jego pomocą rysujemy wykres funkcji `f` w przedziale `[x_min,x_max]` przy dodatkowych opcjach `opt1`, `opt2`, ... Dodatkowe opcje są podawane w formie listy, tzn. w nawiasach kwadratowych.
Przykładowo, jeśli chcemy ograniczyć zasięg wyświetlanych wartości funkcji f, to używamy opcji postaci `[y,y_min,y_max]`. Przy wykorzystaniu tej opcji wykres zawsze będzie się znajdował wewnątrz podanego zasięgu, niezależnie od wartości osiąganych w wykresie. Jeżeli poziomy zasięg nie jest sprecyzowany, będzie on ustawiony jako minimum i maksimum drugiej współrzędnej punktów wykresu. Przykładowo,

```
(%i33)  plot2d(x^2,[x,-3,3],[y,-1,10]);
```



;